Midterm I
CS164, Spring 2003
SOLUTIONS

February 25, 2003

- Please read all instructions (including these) carefully.
- **Write your name, login and circle the section time.**
- There are 7 pages in this exam and 4 questions, each with multiple parts. Some questions span multiple pages. All questions have some easy parts and some hard parts. If you get stuck on a question move on and come back to it later.
- You have 1 hour and 20 minutes to work on the exam.
- The exam is closed book, but you may refer to your two pages of handwritten notes.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. We might deduct points if your solution is far more complicated than necessary. Partial solutions will be graded for partial credit.

NAME and LOGIN: _____

SID or SS#: _____

Circle the time of your section: 11:00 12:00 1:00 2:00 3:00 4:00

Problem	Max points	Points
1	15	
2	15	
3	35	
4	35	
TOTAL	100	

1 Context-Free Grammars (15 points)

Consider the following grammar.

$$\begin{array}{lcl} S & \rightarrow & S\ S \\ S & \rightarrow & 0 \\ S & \rightarrow & \epsilon \end{array}$$

a) Write a regular expression for the language that is accepted by this grammar.

Answer:

$$0^*$$

b) Give at least one reason why this grammar is not LL(1).

Answer:

- It is ambiguous
- It is left-recursive

c) Which single production can you remove from this grammar to make it LL(1)? Does this transformation change the language recognized by the grammar?

Answer:

$$S \rightarrow S\ S$$

The remaining grammar is trivially LL(1), but it describes a different language.

d) Is it possible to make the original grammar LL(1) by adding one production? If your answer is yes, show one such production. If your answer is no, explain why not.

Answer: No, adding productions does not eliminate ambiguity nor the left-recursion.

e) Write an LL(1) grammar for the language recognized by the original grammar.

$$S \rightarrow \epsilon \mid 0\ S$$

2 LL Parsing (15 points)

Consider the following grammar over the alphabet $\{c, d, e\}$:

$$\begin{array}{lcl} S & \rightarrow & A\ B\ A \\ A & \rightarrow & B\ c \mid d\ A \mid \epsilon \\ B & \rightarrow & e\ A \end{array}$$

a) Fill in the table below with the First and Follow sets for the non-terminals in this grammar:

	First	Follow
S	d, e	$
A	ϵ, d, e	e, $, c, d
B	e	c, d, e, $

b) Fill in the column headings, and the row corresponding to A in the predictive (LL1) parsing table for this grammar.

	c	d	e	$
A	ϵ	ϵ d A	ϵ B c	ϵ

c) Is this grammar LL(1)? Explain briefly why or why not.

Answer: No, because at least one entry in the LL(1) table has multiple productions in it.

3 LR(1) Parsing (35 points)

Consider the following grammar:

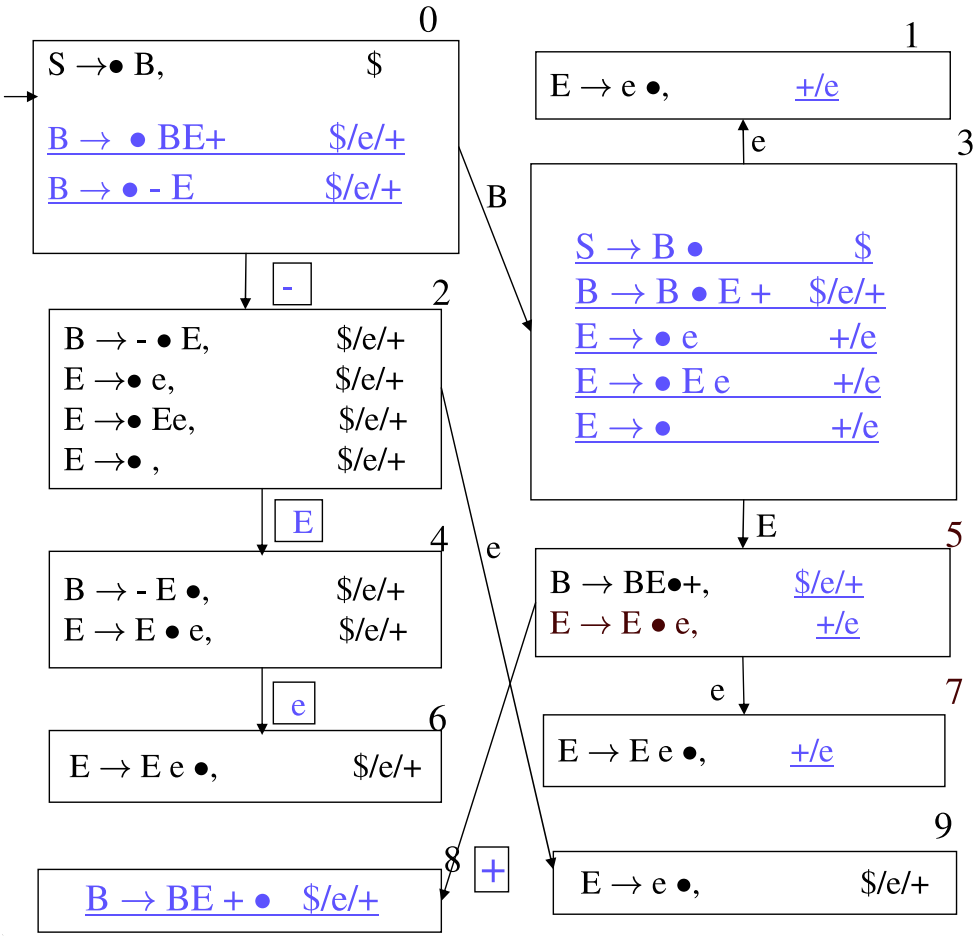
$$
\begin{array}{lcl}
S & \rightarrow & B \\
B & \rightarrow & B\ E\ + \\
 & | & -\ E \\
E & \rightarrow & e \\
 & | & E\ e \\
 & | & \epsilon
\end{array}
$$

Below is a partial DFA for the grammar above. Notice we did not have to add a new start state as is usually done for LR(1) parsers.

a) Complete state 0 by performing closure on the item listed. Don't forget to include lookahead symbols for the new items.

b) Fill in all elements of states 3 and 8, and the lookahead items in states 1, 5 and 7.

c) Fill in the missing transition labels (4 of them).



d) Fill in the following table the **reduce** productions and lookahead tokens for the given states. Write "not a reduce state" if that state does not have a reduction action.

State	Production	Lookahead Symbols
0	not a reduce state	
2	$E \to \epsilon$	\$/e/+
4	$B \to - E$	\$/e/+

e) For each state with a shift-reduce conflict, list the state, the lookahead token and the production that are in conflict. Write "none" if there are no shift-reduce conflicts.

Answer:

2	$E \to \epsilon$	e
3	$E \to \epsilon$	e
4	$B \to - E$	e

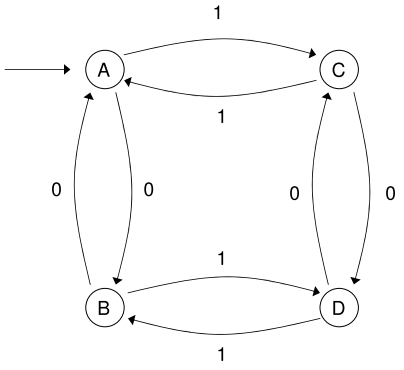
f) For each state with a reduce-reduce conflict, list the state, and the conflicting productions. Write "none" if there are no reduce-reduce conflicts. Answer: none

g) For each of the following configurations of the LR(1) parser, say what action is taken by the parser (one of "shift", "reduce with production ...", "error" or "accept") and write the next configuration (except in the case of an accept or error). You can assume that in the case of a shift-reduce conflict the parser chooses to shift.

Configuration	Action	Next Configuration
$- \triangleright e + \$$	shift	$-e \triangleright \$$
$BEe \triangleright e + \$$	reduce $E \to E\ e$	$BE \triangleright e + \$$
$Be \triangleright \$$	error	
$B \triangleright \$$	accept	

4 Regular Expressions and Finite Automata (35 points)

Consider the following finite automaton over the alphabet $\{0, 1\}$. A is the starting state but so far there are no accepting states.



a) Circle the correct answer: this automaton is a DFA not a DFA

Answer: a DFA because it does not have ϵ transitions or multiple transitions for the same label.

b) Which states should be made accepting in order for this automaton to accept the language of strings with zero or an even number of 1?

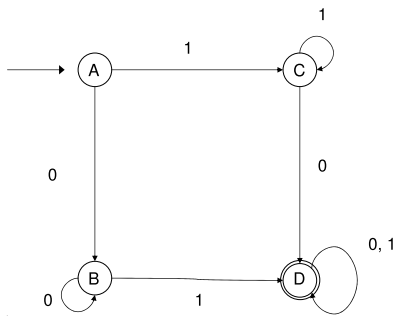
Answer: A, B

c) Which states should be made accepting in order for this automaton to accept the language of strings with odd length?

Answer: B, C

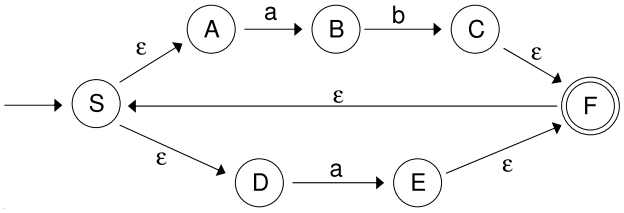
d) Draw a deterministic finite automaton (DFA) that recognizes the language over the alphabet $\{0, 1\}$ consisting of all strings that contain a least one 0 **and** at least one 1. Remember to mark the start state.

Answer:

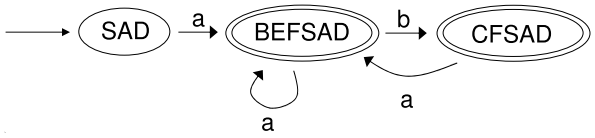


State A represents, "no 0 or 1 seen so far", state B represents "seen at least one 0 but no 1", state C represents "seen at least one 1 but no 0", and state D represents "seen both 0 and 1".

e) Consider the following NFA over the alphabet $\{a, b\}$. Convert this NFA to a DFA. For each DFA state write the set of the NFA states that it corresponds to. Remember to mark the start state.

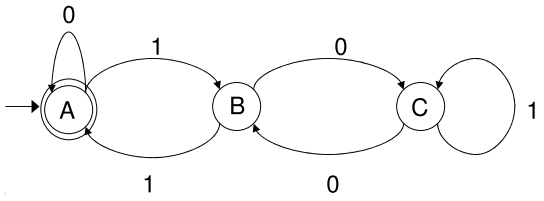


Answer:



f) Draw a deterministic finite automaton for the language over the alphabet $\{0, 1\}$ that consists of the binary numbers that are divisible by 3 (allowing leading 0's). The empty string is also part of the language. **This question is harder that others in the exam. Do not spend a lot of time on it unless you have checked your answers on the other questions.** Hints: There is a solution with 3 states. Make sure you verify your answer on the binary representation of a few numbers like 3,6,9,15.

Answer:



State 0 represents multiples of 3, state B multiples of 3 plus 1, and state C multiples of C plus 2. Here is how to to find out where the arrow 0 from state B goes: the numbers at B can be written as $3k + 1$. Adding a 0 at the end mutiplies the number by 2, resulting in $6k + 2$, which is a multiple of 3 plus 2, thus it goes to state C.